CSCI 6444 Big Data and Analytics

Class Project #2

Report

**Text Analytics in R**

Name: Tianyu Yang (G38878678)

Zhijun Xia (G23682615)

Date: 7/14/2019

Instructor: Stephen Kaisler

Term: 2019 Summer

[**Introduction**](#_r9jkgfr3css) **2**

[**Result**](#_olnxhfjd54dn) **3**

[2.a Lecture 4 Function Demonstration](#_xuq1s6wq0g7) 3

[2.b Longest sentences](#_cy52m4m9xxui) 15

[2.c Dendrogram and WordCloud for each paragraph](#_k3aq9iaf9lu1) 17

[2.d Longest word and sentences](#_5msd0optixrg) 24

[2.e WordNet Demonstration](#_6u256go4g5nc) 27

[2.f Word Frequency Analysis](#_bx1y36z36qb9) 30

[2.g Bigrams and Trigrams](#_d3vwmkorvpgh) 36

[2.h Three Packages Demonstration](#_1vkk6dnv0i7b) 38

[2.h.1 Three packages](#_lr7577b3cmfr) 38

[2.h.2 The theme of this article](#_1gtgnb9w590x) 41

[**Word Search**](#_xoxtjne6x2ra) **41**

[3.1 Example for the word “efficacious”](#_goes9biklb9j) 43

[3.2 Example for the word “materials”](#_jypc4u7rr4k5) 44

[3.3 Example for the phrase “a host of”](#_spvb2bz9o7yd) 45

[**Conclusion**](#_cla7kl72xack) **46**

# Introduction

In project 2, our group tried to use what we learned from the class to do the text analytics in R language. The data set we used is Dream.txt written by Henri Bergson, which is a popular book and contains 54833 characters.

In this project, we will use several useful packages in R, such as textreuse, wordnet, zipfR, corpusTools, stringi, corpustools, quanteda and so on to generate 8 deliverables (a to h) from the text. The result will be displayed in the Result Section of this report.

Furthermore, we will write R functions to search through the documents to find three specific words or phrases, and display the result in the Word Search Section of this report.

Last but not the least, we will give out the conclusion of this project, which will give a summary of this project and show what we learn from this project and knowledge about data science in the Conclusion Section. All the R codes will be zipped together with this report as the name Group-2-Project-2.zip.

# Result

## 2.a Lecture 4 Function Demonstration

Create a VCorpus:



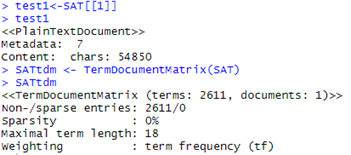
Inspect the VCorpus:



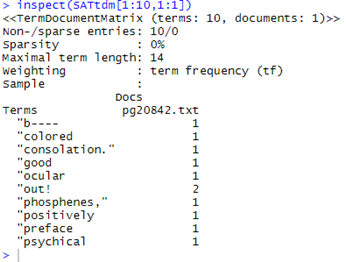
VCorpus Structure:

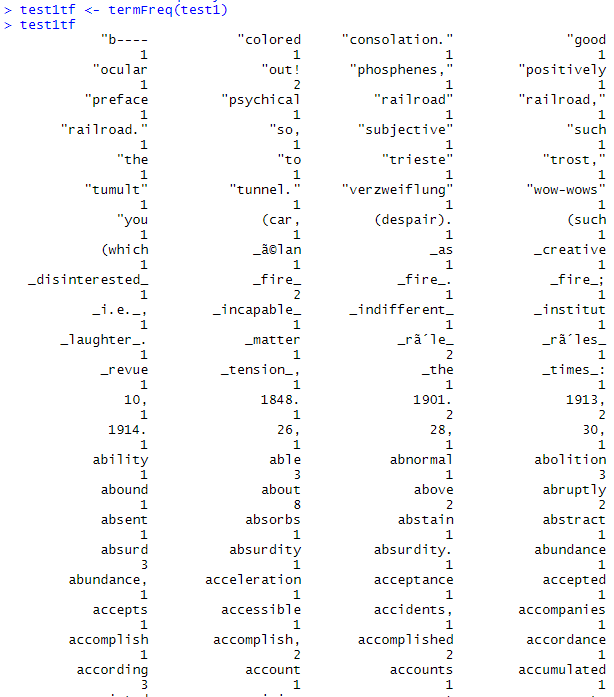


Extract a document from SAT:



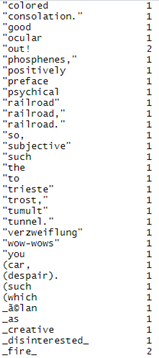
Inspecting the TDM:



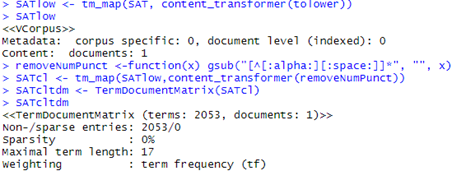
Document Term Frequency:

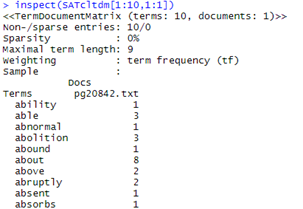
...

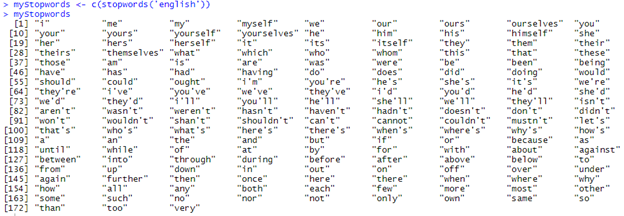


...

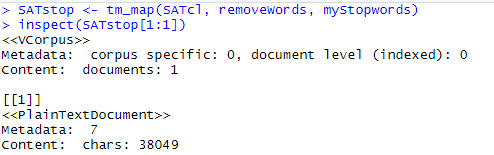
Corpus Management:



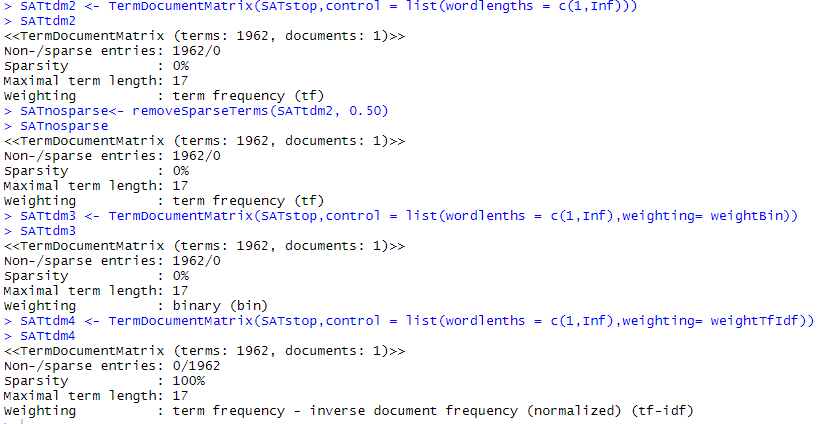




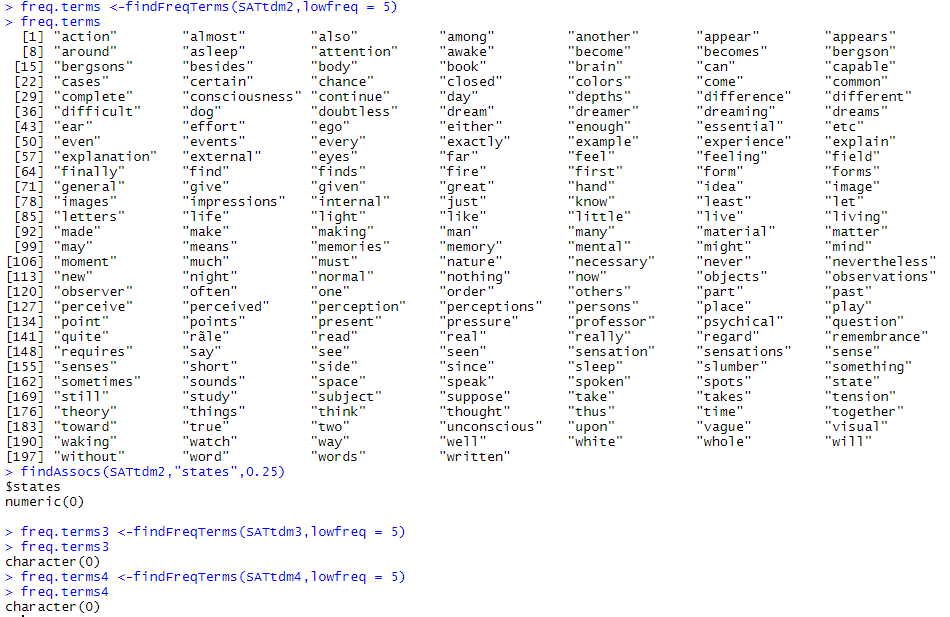
Removing stop words:



Remove sparse terms:



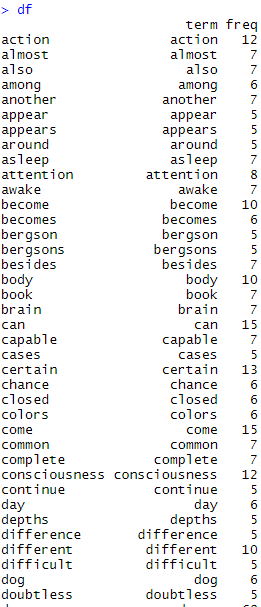
Finding Frequent Words:



Term Frequency:

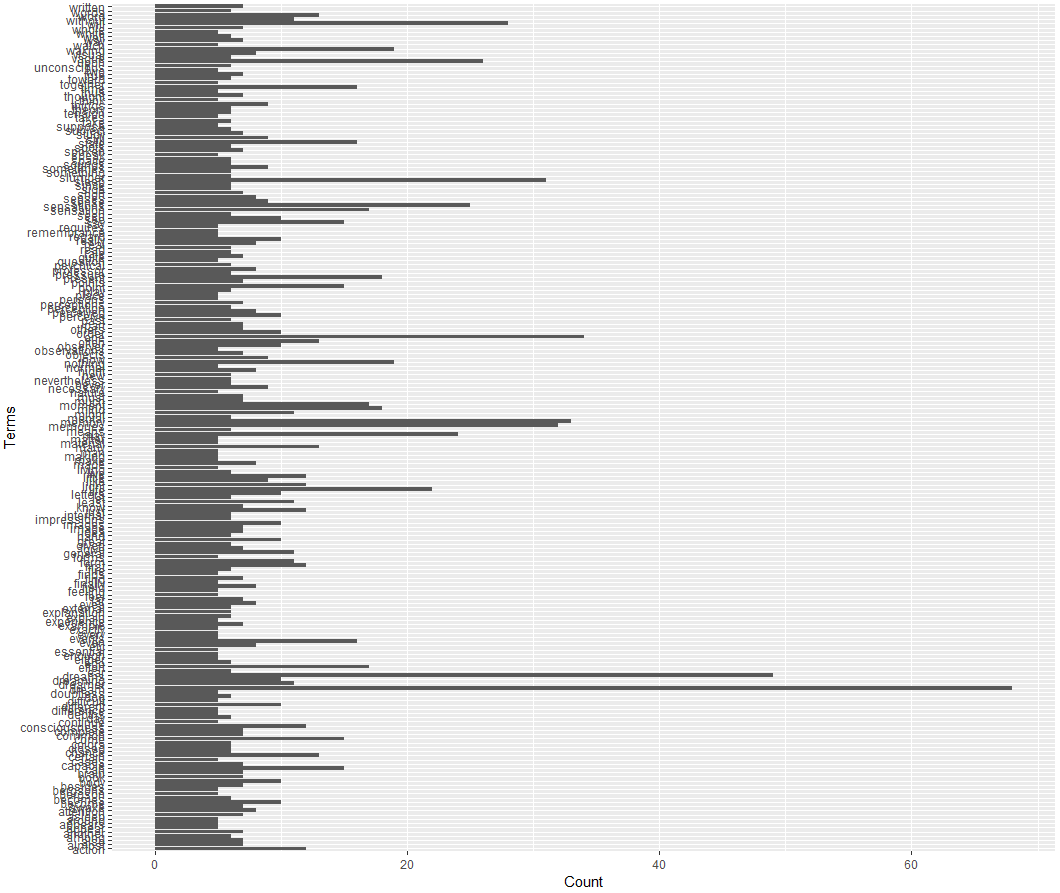


…

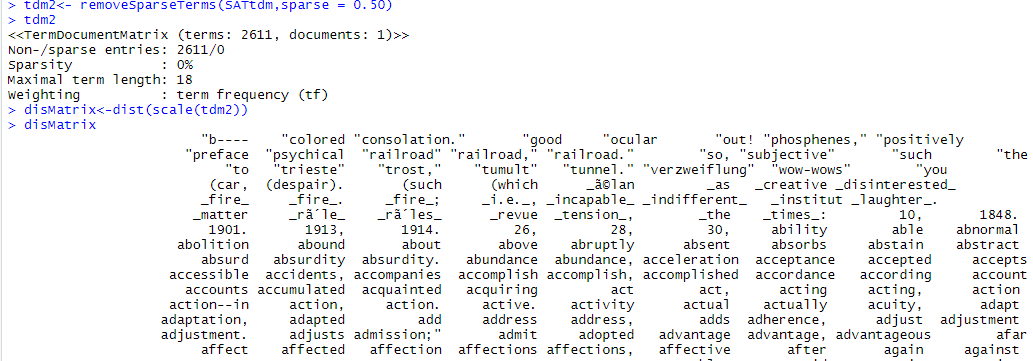
...

Plot graph:



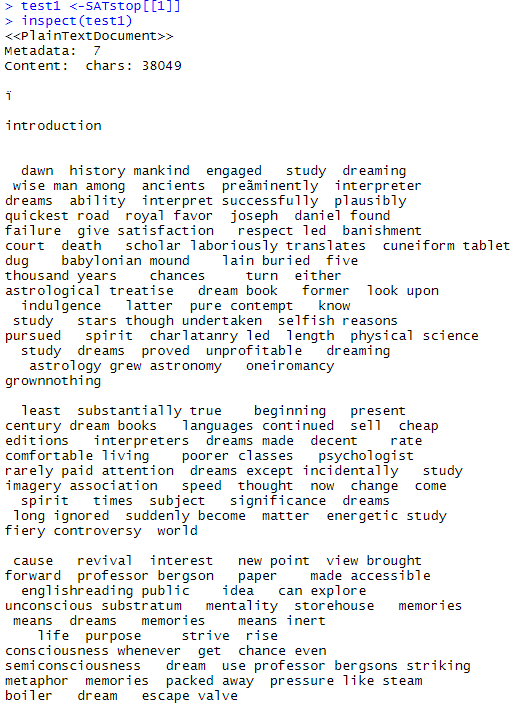


Clustering Terms:

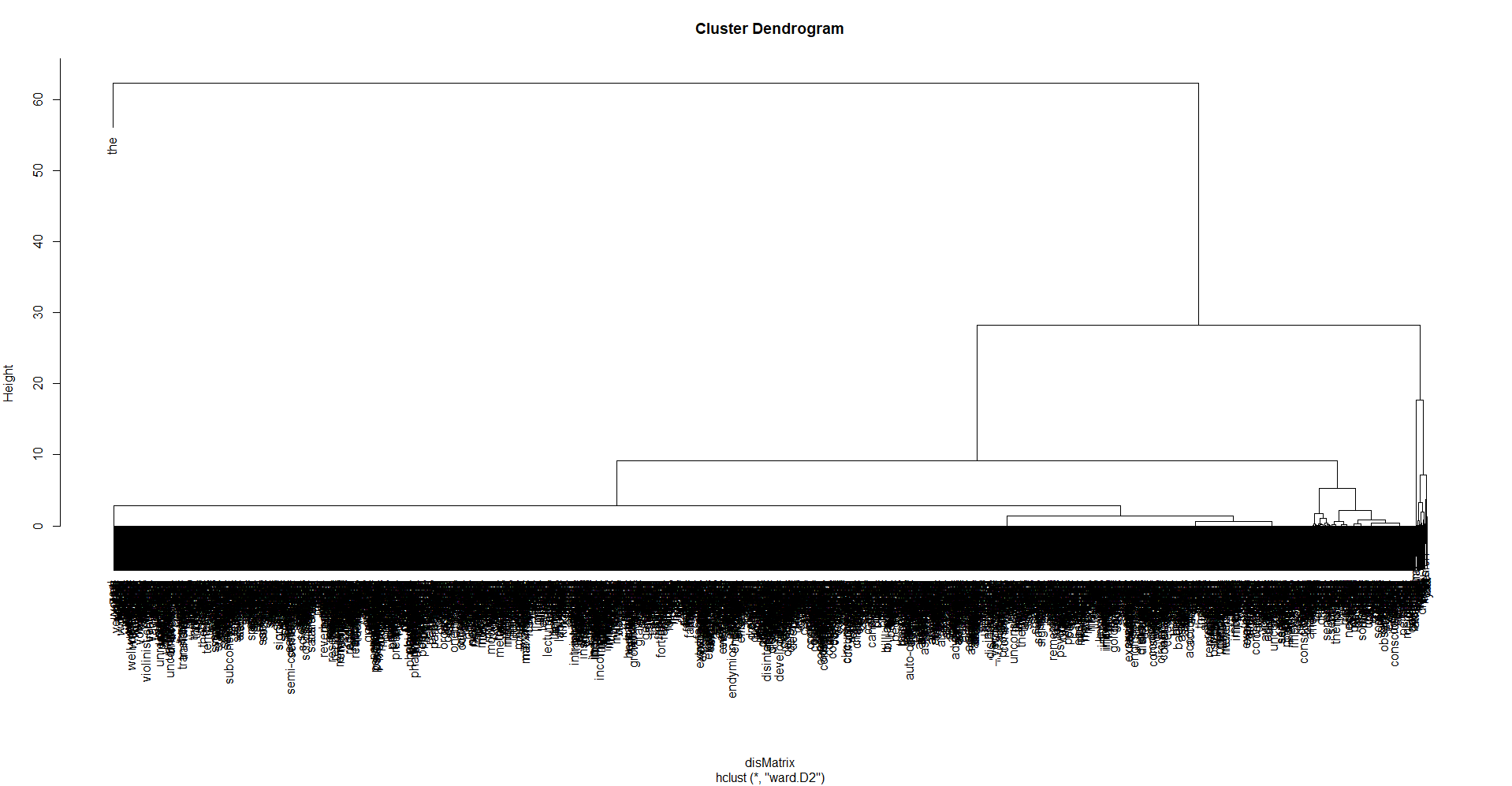


…

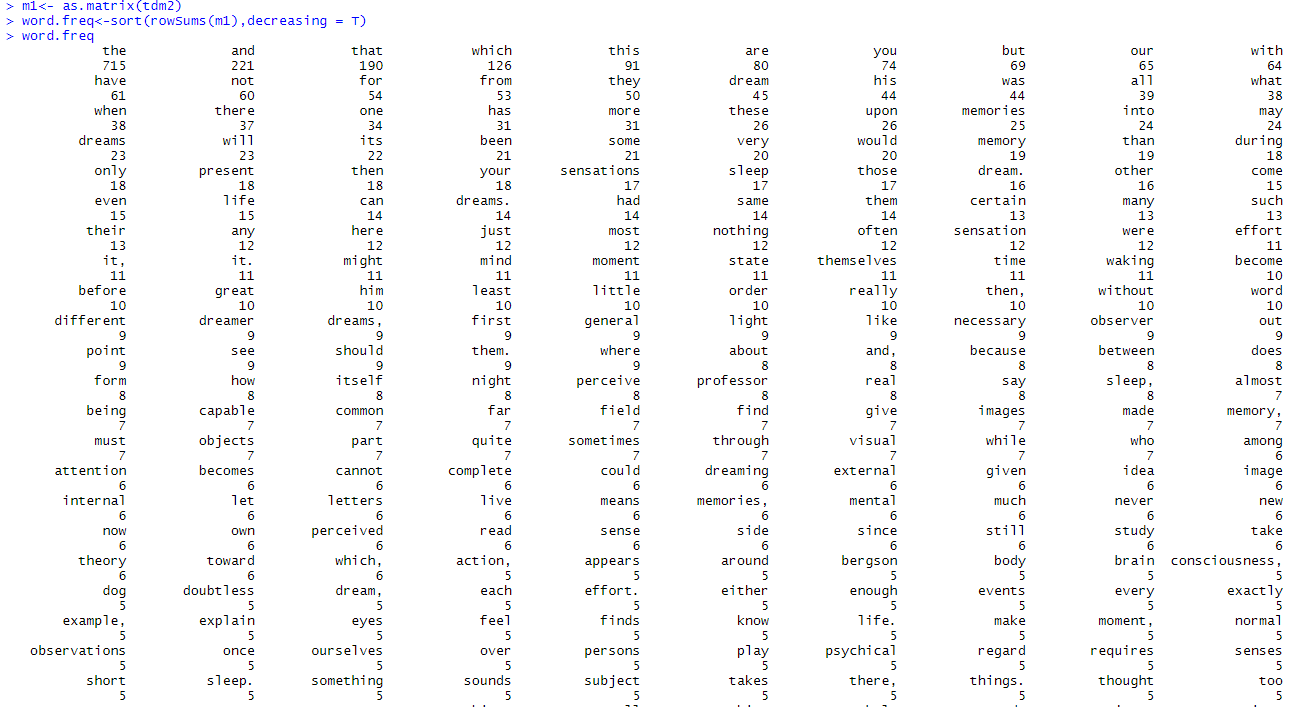
Finding Informative Words:

...

Clustering Terms via Dendrogram:

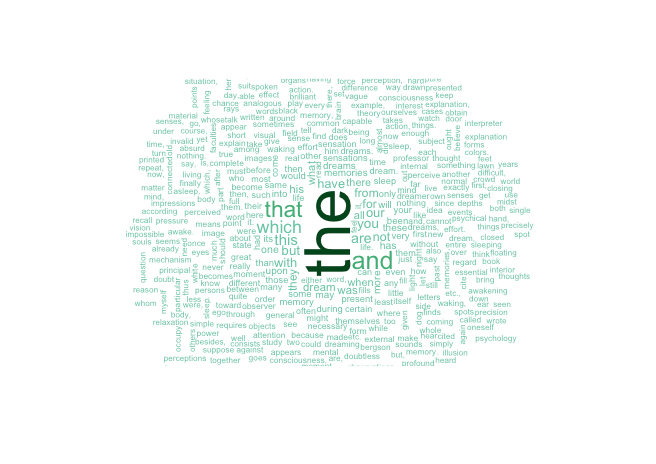


Word Cloud:

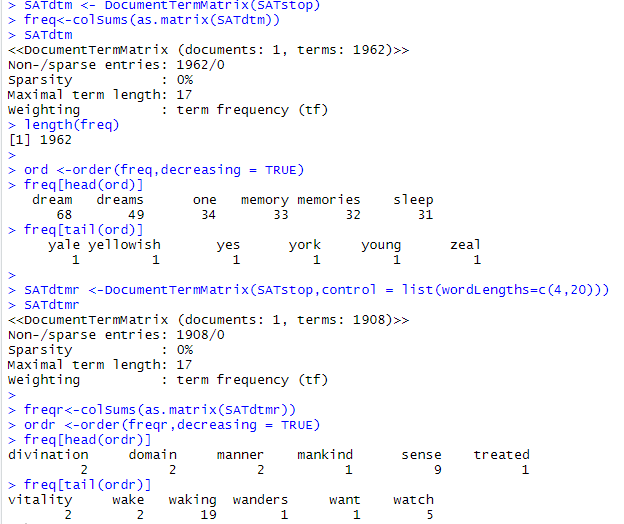


…

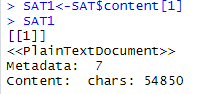


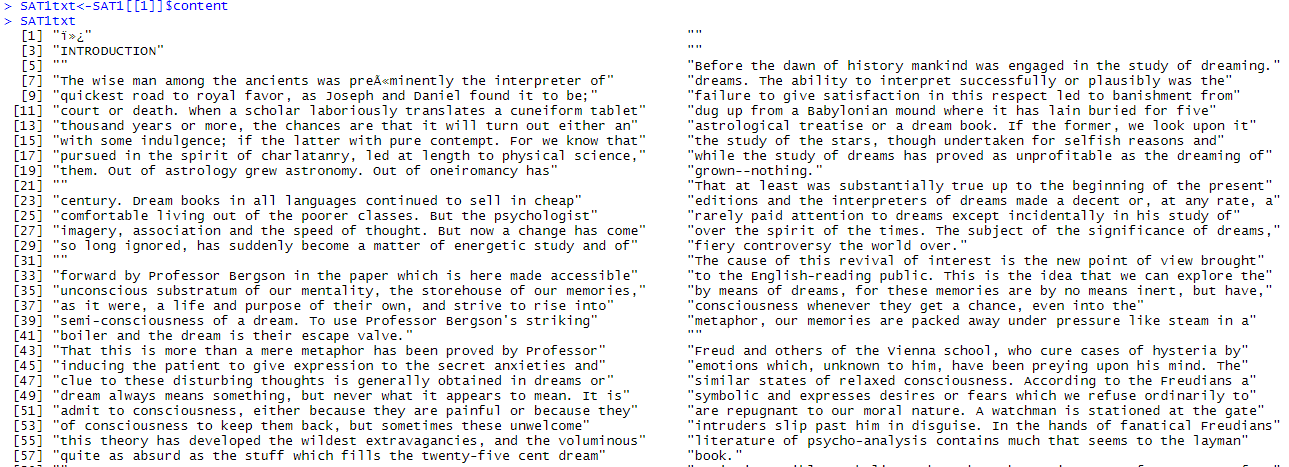


Frequency Analysis:



Text Analysis:

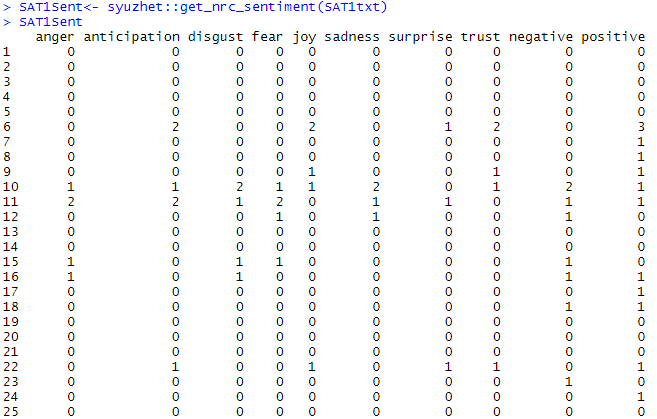




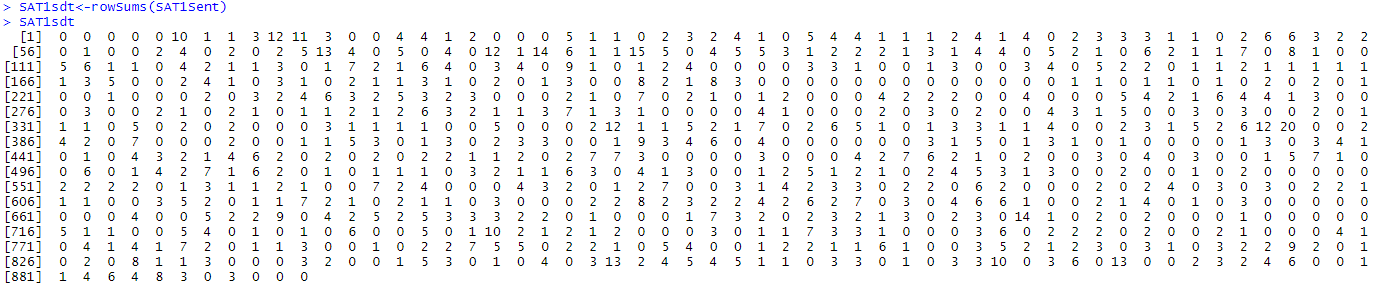
…

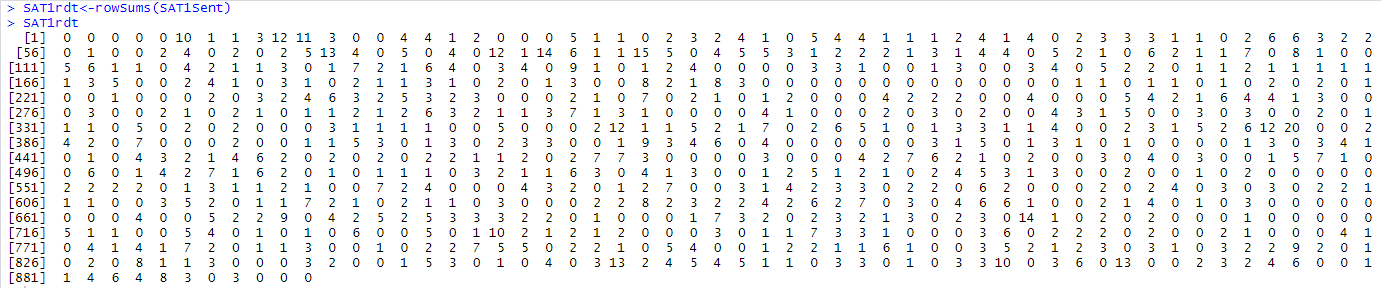


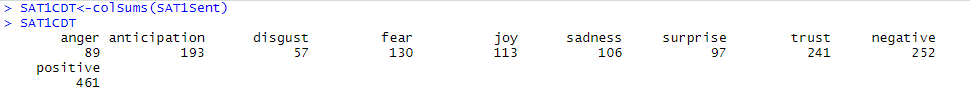
Text Analysis: Sentiment Analysis:



…



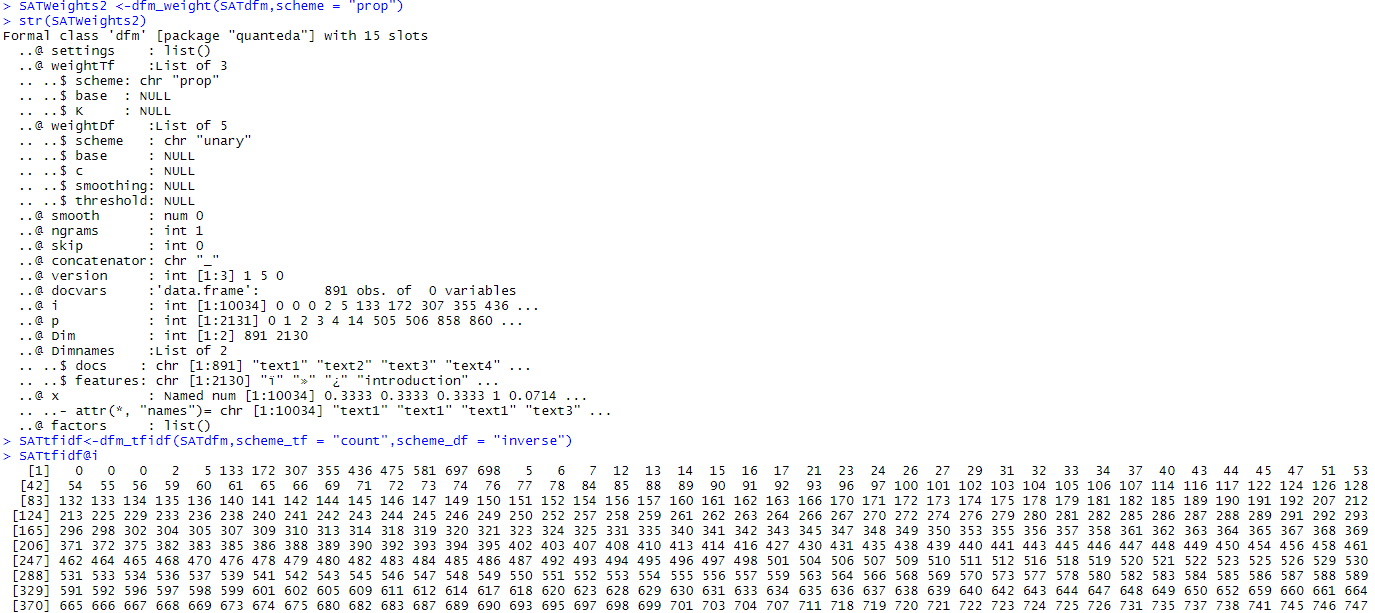




Text Weighting:



…

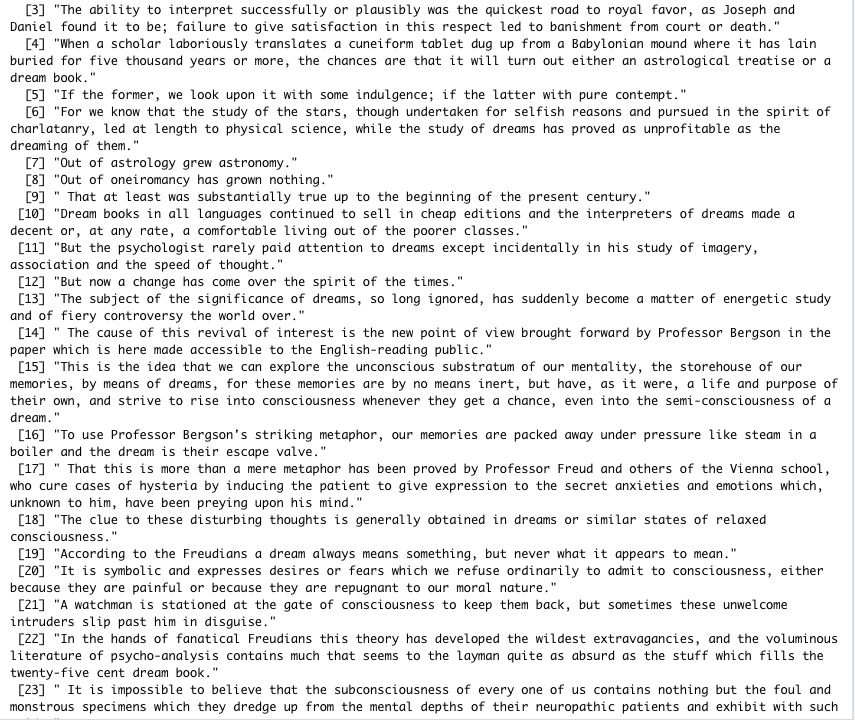


…

## 2.b Longest sentences

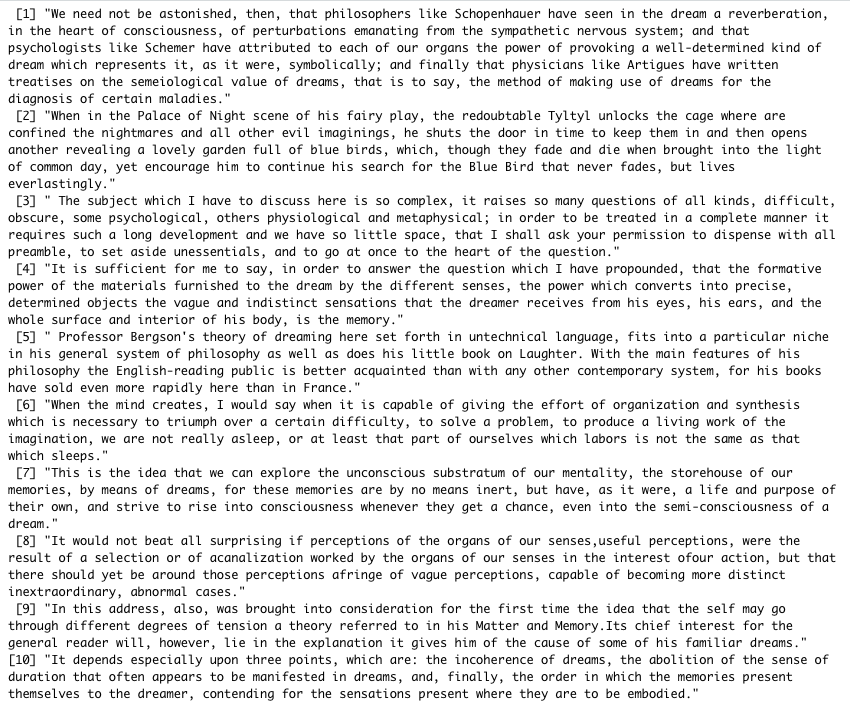
(1) To obtain the longest sentences in the text, firstly we need to get the vectors contains every sentence in the text, shown as below.





(2) Write two R functions to realize the function. The first one is to obtain the index of the sentence that contains the most words. The second one is to obtain n longest sentences in the text. It is first to get the longest sentences and then remove it from the text. Repeatedly n times, then we can get n longest sentences. When n = 10, it is shown as below:





## 2.c Dendrogram and WordCloud for each paragraph

To obtain the dendrogram and wordCloud for each paragraph, we write two functions to create dendrogram and wordCloud.

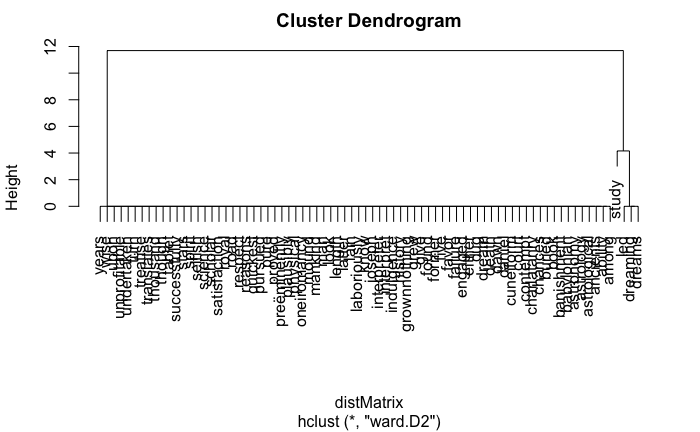


Before plot the figure, we need to finish the data cleaning. Firstly, we remove numbers and punctuation in the text. Then we clear all stopwords in ‘English’. Finally we list all the word tokens in lower level.

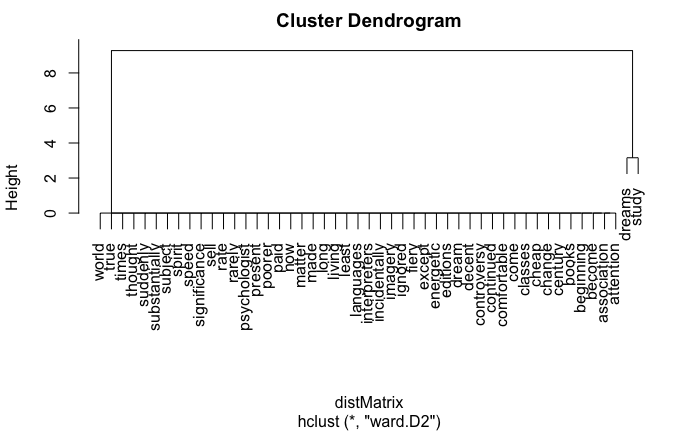
From 2.d, we separate the text into paragraphs. Therefore, we use for-loop to display the dendrogram and wordCloud for all the paragraphs in the text. Because of the limitation of the space of the report, we only display five of each as examples. For there is a limitation for the length of the paragraph, therefore we choose 1-4, 7 as examples.

1. Dendrogram:

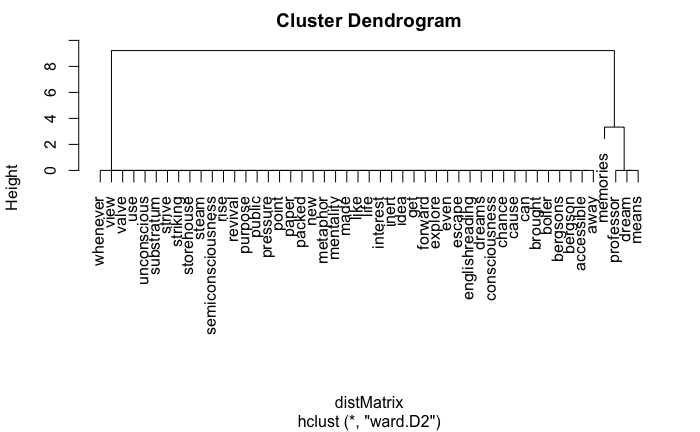
Paragraph 1:



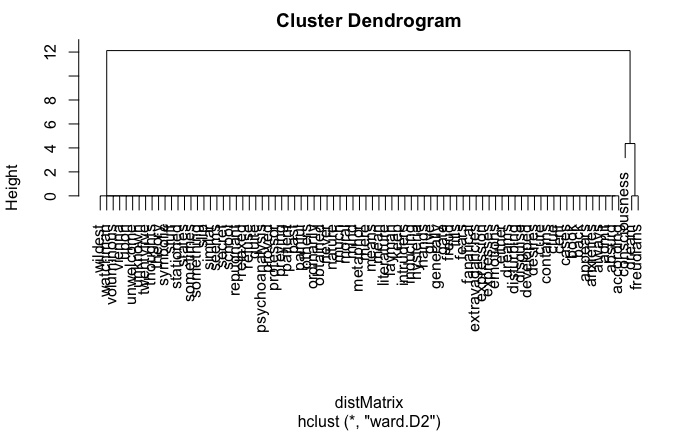
Paragraph 2:



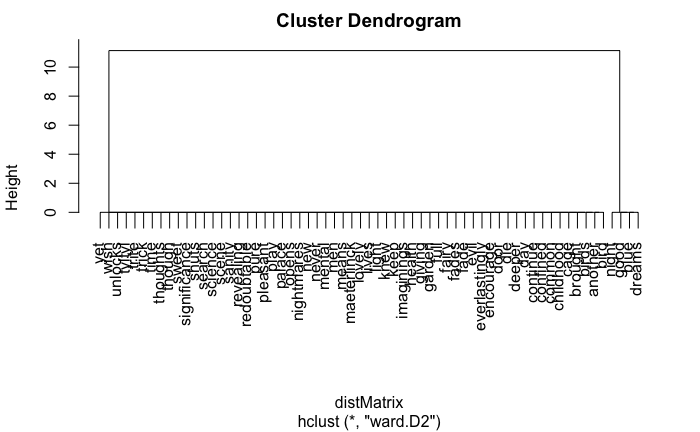
Paragraph 3:



Paragraph 4:

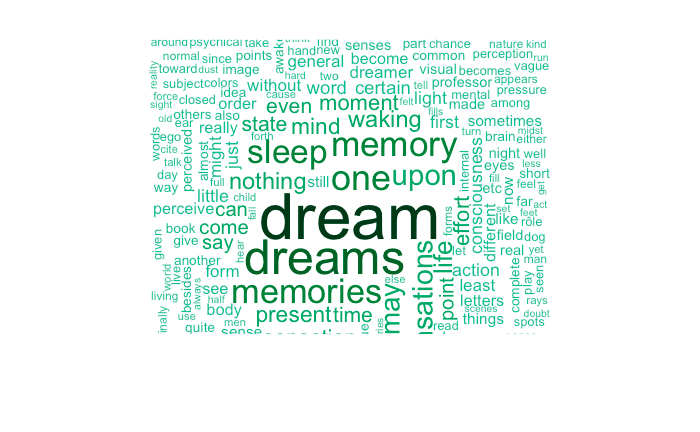


Paragraph 7:



1. WordCloud:

Paragraph 1:



Paragraph 2:



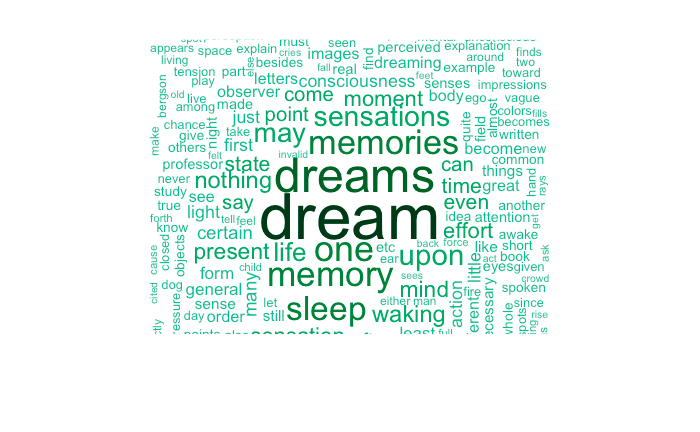
Paragraph 3:



Paragraph 4:



Paragraph 7:



## 2.d Longest word and sentences

In order to find the longest word, the longest sentence, the length of the longest sentence and the shortest sentence in each paragraph, firstly we need to separate the text into paragraphs.

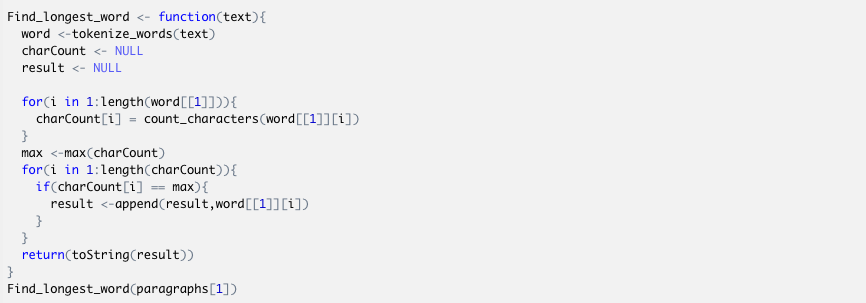


This function is used to generate a vector which contains the paragraphs of the input text. To locate each of the paragraphs, we use “\”\”” to separate texts into paragraphs. If the number of words no less than 7, we save it into a vector.

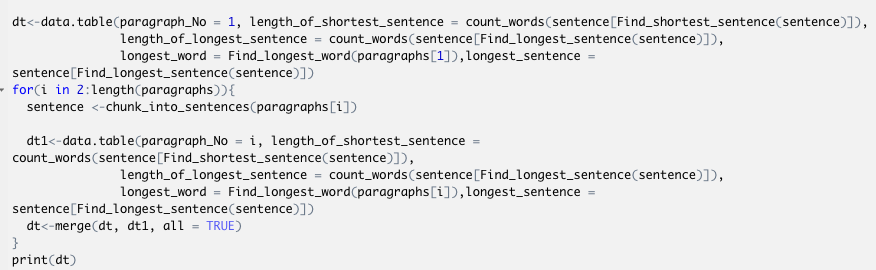
From 2.b, we can easily find the longest sentence. Similarly, we use the same method to find the shortest sentence.



To find the longest word, we count the character number for each word and find the word with the most characters.

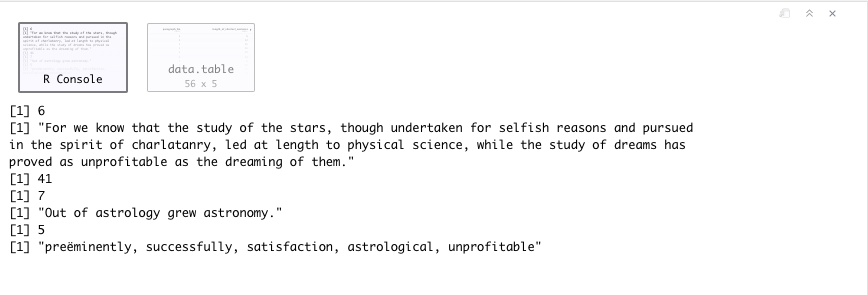


After that, we use a datatable to display the required variable in all paragraphs.



The result is shown as below:

1. As an example, I display the deliverariables in the first paragraph.



It shows that in the first paragraph, the longest sentence is the 6th sentence and it contains 41 words.

*"For we know that the study of the stars, though undertaken for selfish reasons and pursued in the spirit of charlatanry, led at length to physical science, while the study of dreams has proved as unprofitable as the dreaming of them."*

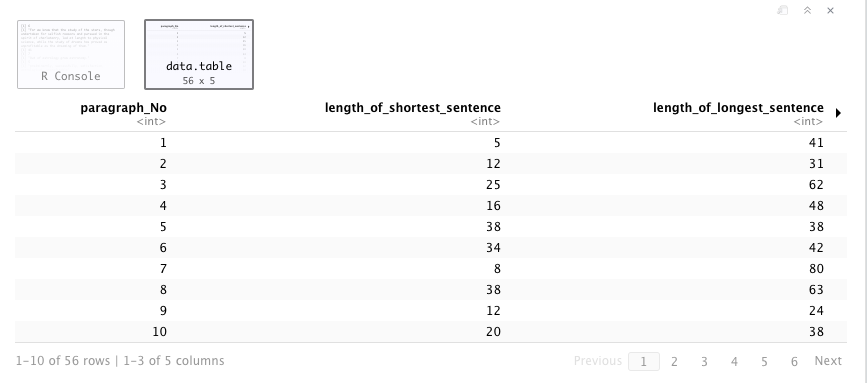
The shortest sentence is the 7th sentence and it contains 5 words.

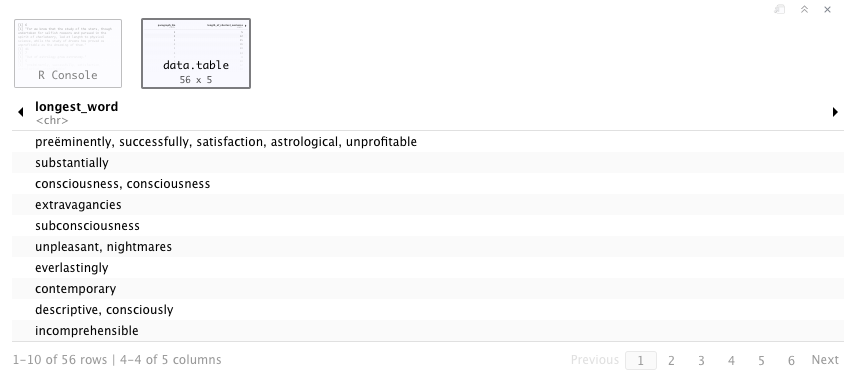
*"Out of astrology grew astronomy."*

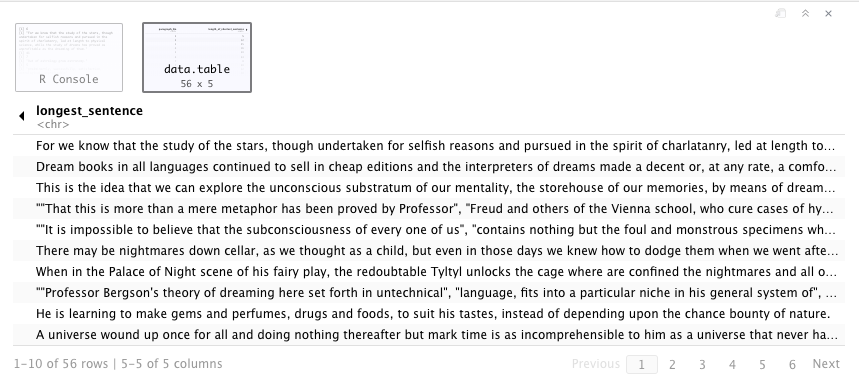
There are totally fix words containing the most characters(12 characters).

*"preëminently, successfully, satisfaction, astrological, unprofitable"*

(2) Here is the table which display the required deliverariables. Totally there are 56 paragraphs in the text.

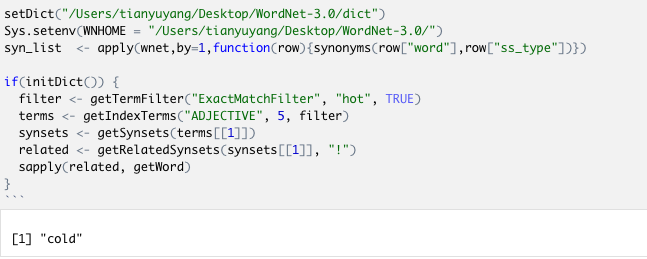




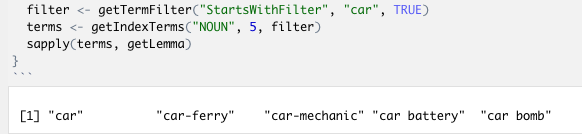


## 2.e WordNet Demonstration

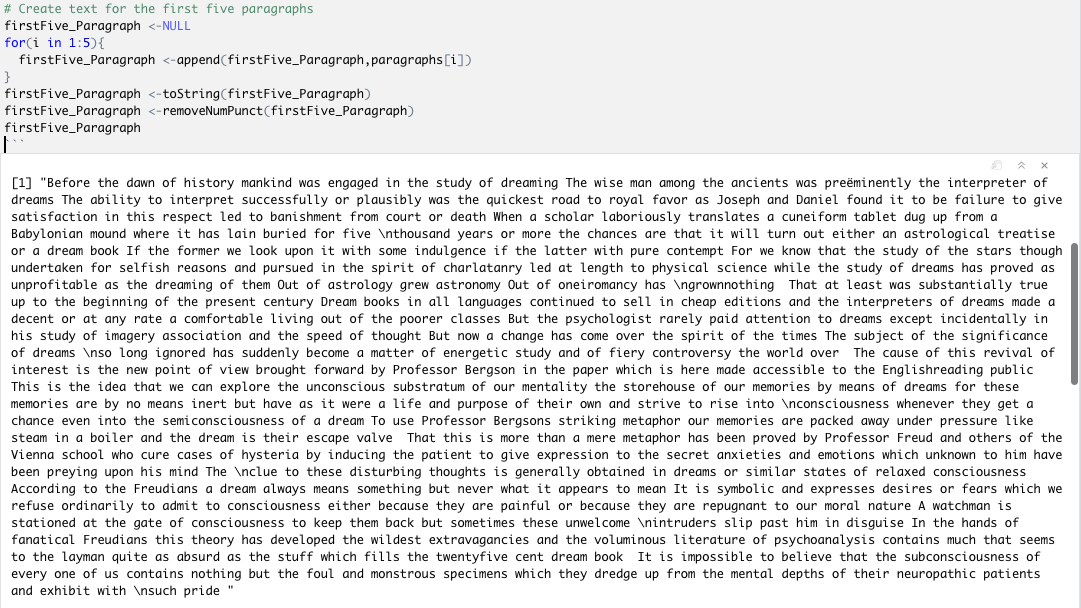
“WordNet” package is used to get a synonym of a word from the dict. Here is an example. The antonym of “hot” is cold.



This is to get the nouns related to the word “car”.



To get verbs and nouns in the first five paragraphs which contain no less than 5 characters, we firstly need to get the first five paragraphs. We use the paragraph function in 2.d.



Get the words that contain no less than 5 characters and make them as a list.



Use synonyms(word, pos) function in wordnet to judge the part of speech of a word. pos is the input part of speech, including “NOUN”, “VERB”, “ADVERB” and “ADJECTIVE”. To get all the verbs and nouns. We set pos as NOUN/VERB and judge the length of synoyms(word, pos). If the length is 0, it is not a noun/verb. If the length is non-zero, it is a noun/verb.

The noun list for the first five paragraphs that contain no less than five characters is:



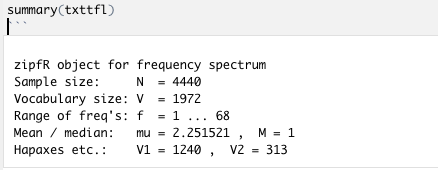
The noun list for the first five paragraphs that contain no less than five characters is:

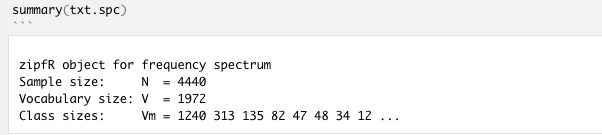


## 2.f Word Frequency Analysis

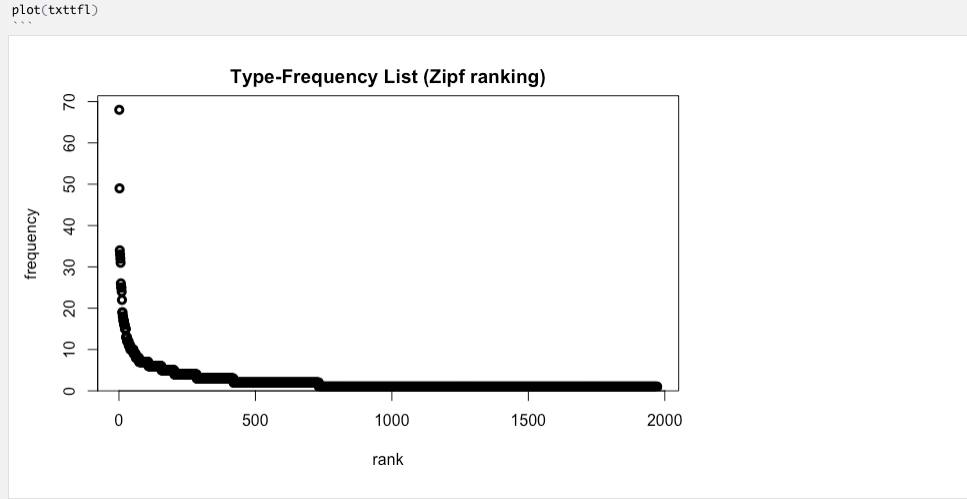
To use zipfR package to analyze word frequency, we firstly need to obtain the right data structure for the input word frequency.

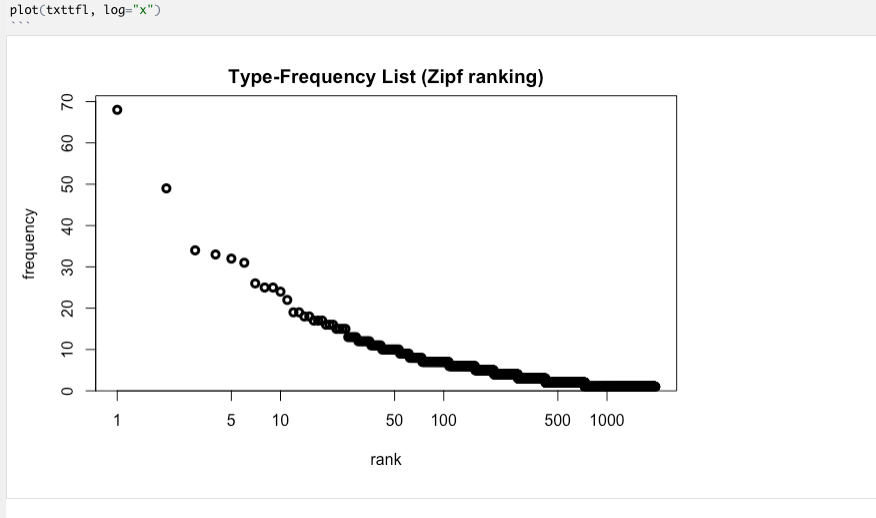


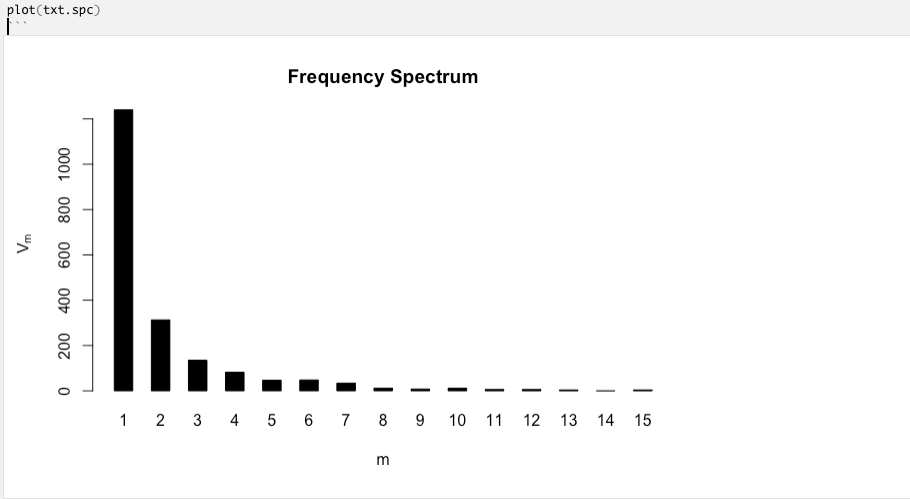


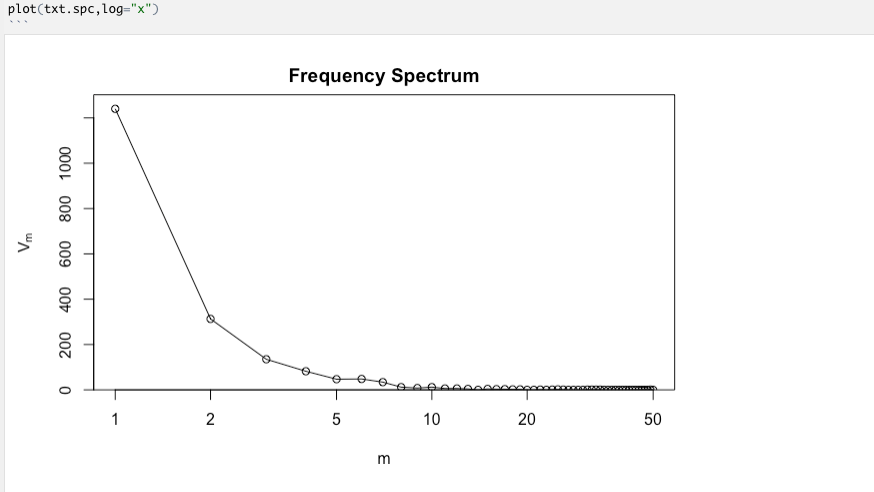


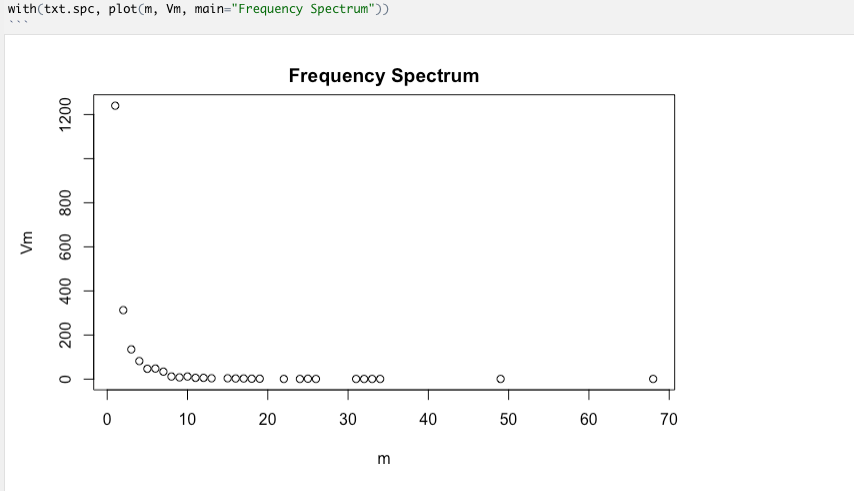
1. Plot the spectrum



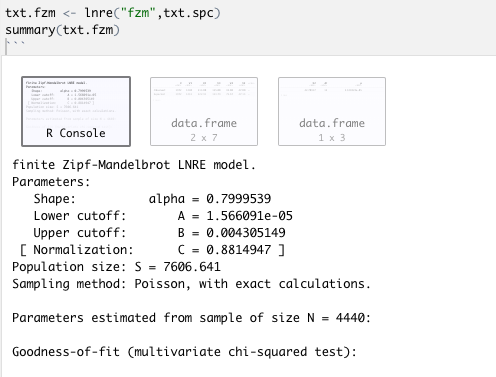


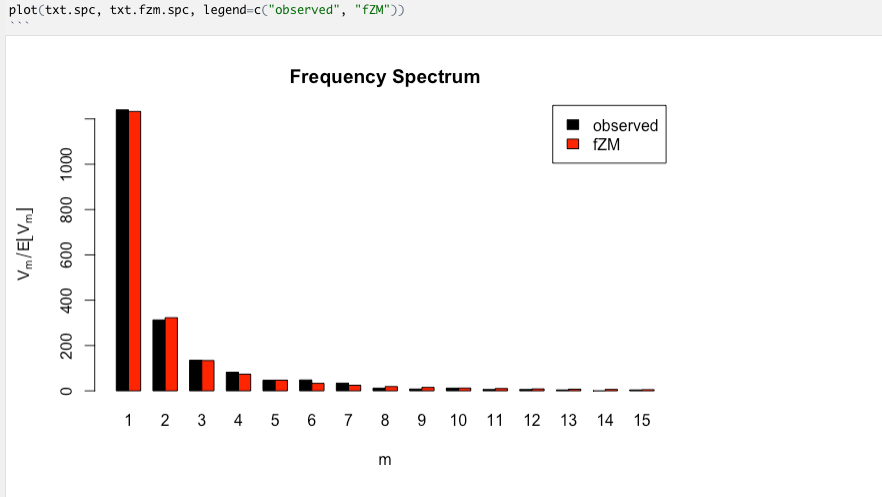




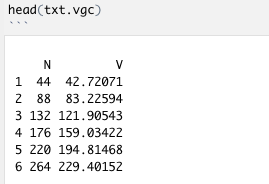


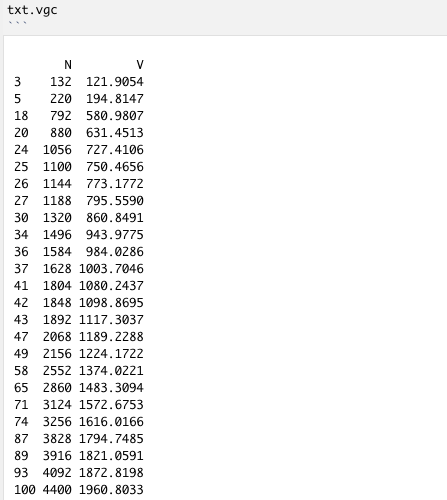
(2) Estimating V and other quantities at arbitrary sample sizes

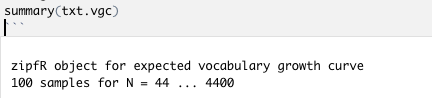


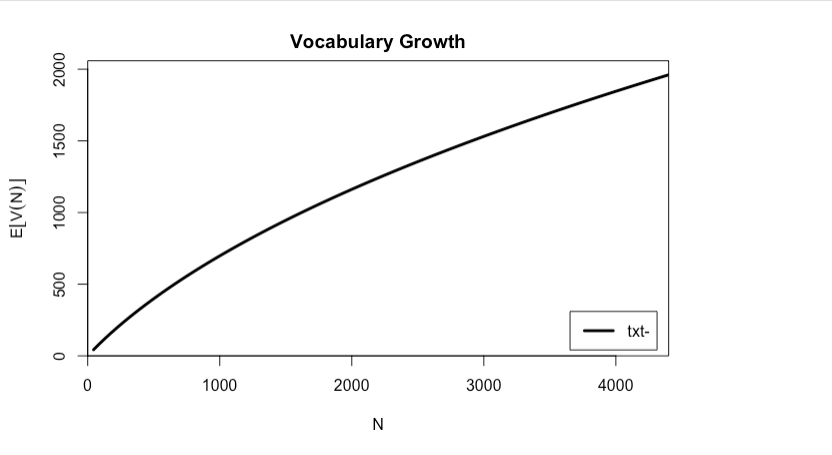


(3) Vocabulary growth curves

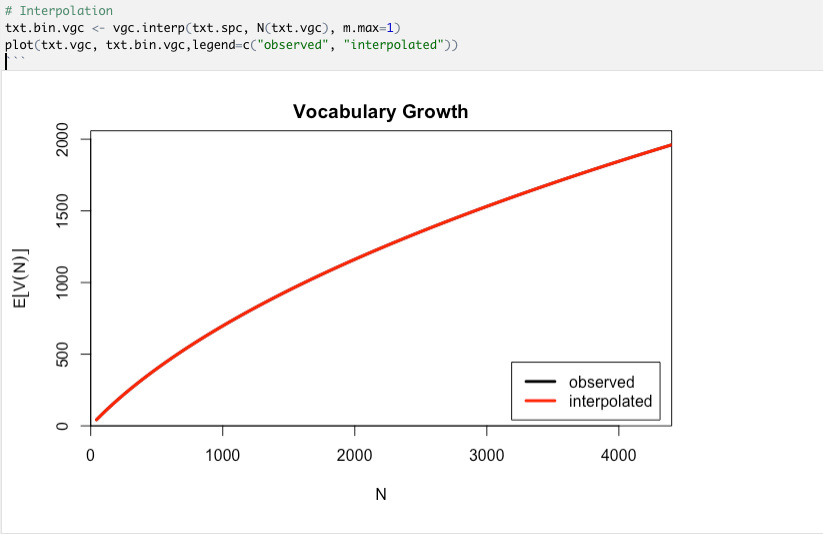






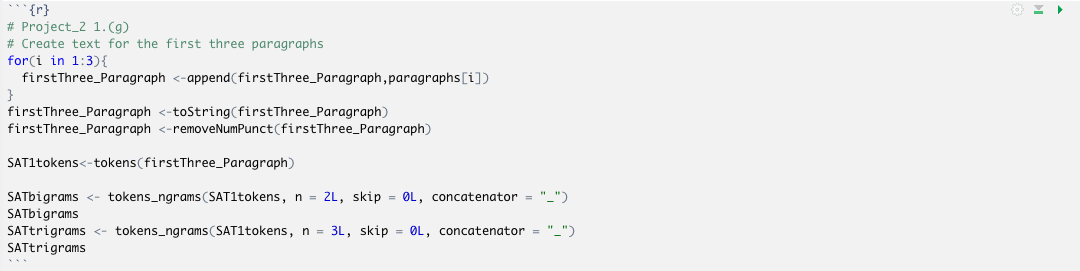


(4) Interpolation



## 2.g Bigrams and Trigrams

To obtain bigrams and trigrams in the first three paragraphs, we firstly need to get a string which contains the content of the first three paragraphs and then remove all the numbers and punctuation.



The bigrams are shown as below:



The trigrams are shown as below:



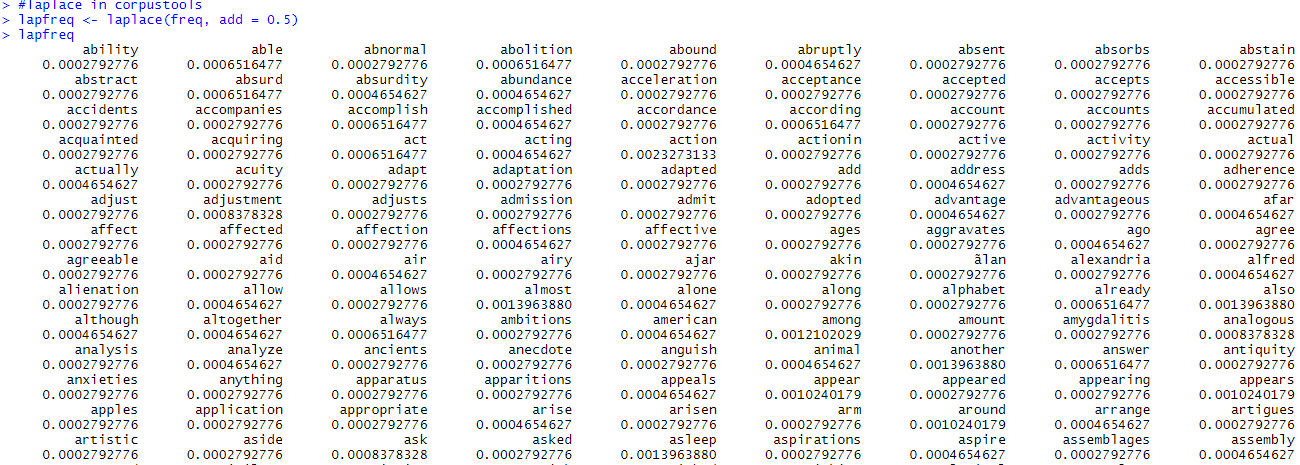
## 2.h Three Packages Demonstration

### 2.h.1 Three packages

(1)CorpusTools:

1.laplace

To laplace (i.e. add constant) smoothing a numeric vector of term frequencies.



2.resources\_path

Get name of the resources location



3.get\_stopwords

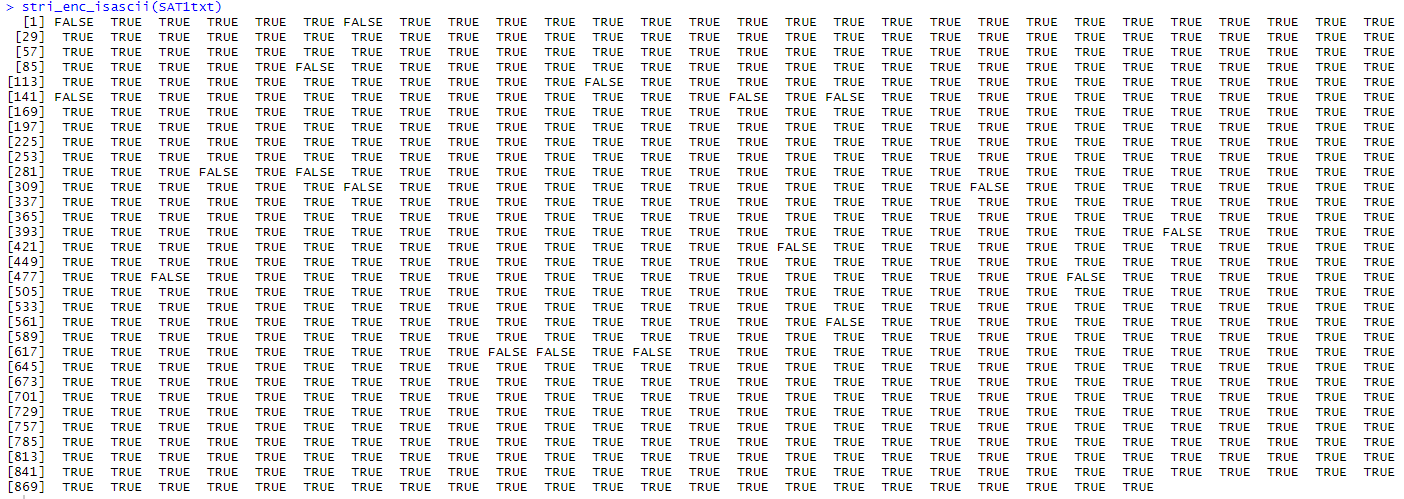
To get a character vector of stopwords



(2)Stringi:

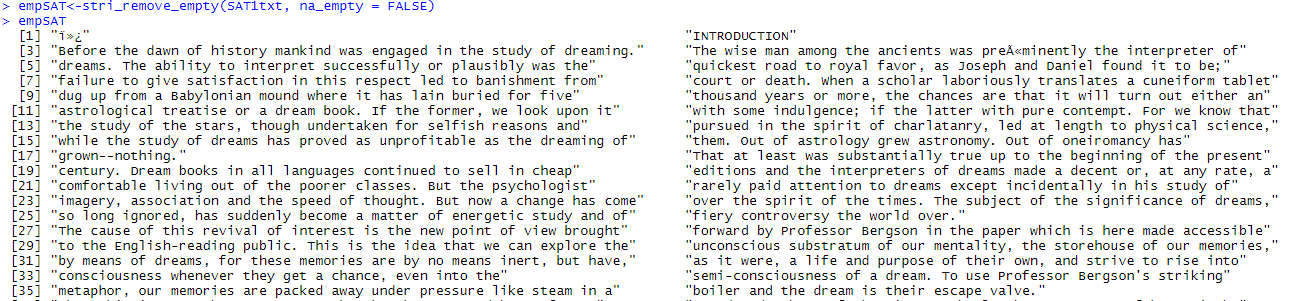
1.stri\_enc\_isascii

To check if a Data Stream Is Possibly in ASCII, the function checks whether all bytes in a string are <= 127.



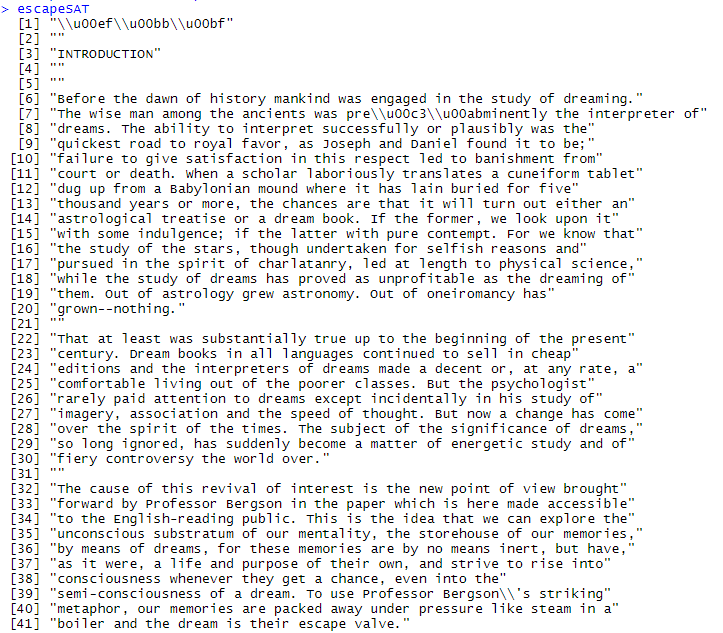
2.stri\_remove\_empty

To remove all empty strings from a character vector



3.stri\_escape\_unicode

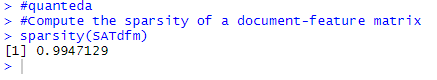
To escapes all Unicode (not ASCII-printable) code points.



(3)Quanteda

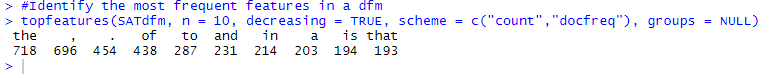
1.sparsity

To compute the sparsity of a document-feature matrix, return the proportion of sparseness of a document-feature matrix, equal to the proportion of cells that have zero counts.



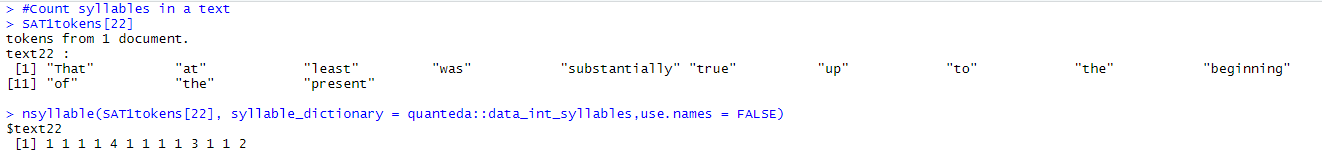
2.topfeatures

To list the most (or least) frequently occurring features in a dfm, either as a whole or separated by document.



3.nsyllable

Returns a count of the number of syllables in texts. For English words, the syllable count is exact and looked up from the CMU pronunciation dictionary, from the default syllable dictionary data\_int\_syllables. For any word not in the dictionary, the syllable count is estimated by counting vowel clusters.



### 2.h.2 The theme of this article

This article is mainly about dreams. From part 2.c above, we can see that the word "dream" appears in almost every paragraph and has the highest frequency. In the introduction part, there is a lot of content about the understanding of dreams by different people of different eras. In the body part, there's a detailed explanation of how outside stimuli of different types can affect the dreaming process. The second most common word is “memory”, the author explores the relationship between dreams and memory. Obviously, memory has a major impact on dreams. Other words that appear frequently and are worth noting are "consciousness", "sensation", "see", "life". We can see that the author is trying to explain dreams and relate them to real-life experience.

# Word Search

To obtain the document number, the line number and word index in the sentence, we write a function to work on it. The input of this function is the searching word/phase in the string and the searching path in the string.

After finishing all the preparations for the variable, we start searching the text.



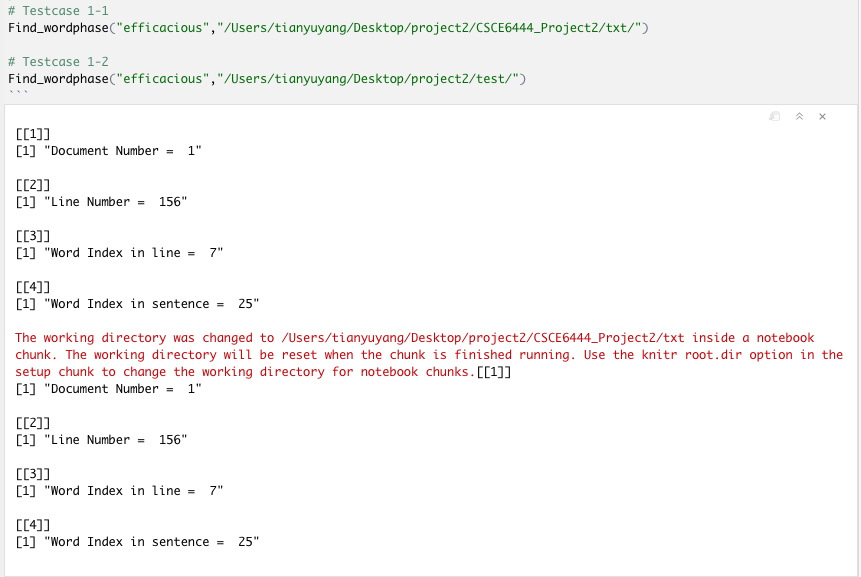
The first for-loop is to find word index in the sentence. Firstly it traversal all the sentences and then obtain the word token from each of the sentences. Then it tries to pair the word or phrase with the tokens. If the searching word/phase is same as one of the tokens, (If it is a phase, it tries to pair both word and order), the word index (if it is a phrase, it will same the first-word index in the sentence) will be saved and output. This searching detects upper and lower case of word’s characters.



Same things we did for the document number and line number. But for these, we do not need to divide sentences.

To test our functions, we give three test cases (including words and phrases more than six characters), in two conditions. We use the entire text as our first condition. For we only have one entire text but we need to test for document number. Therefore, we separate the whole passage into three texts as three documents in a folder as our second condition.

## 3.1 Example for the word “efficacious”



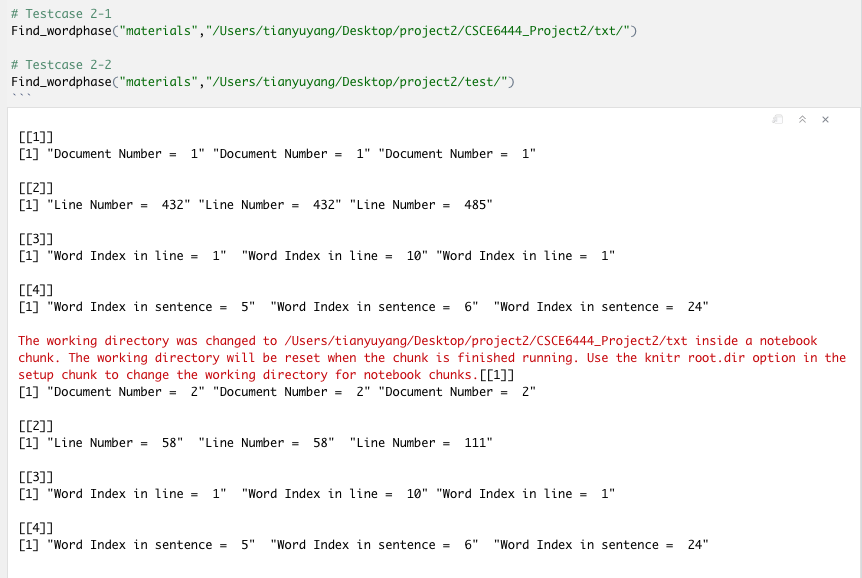
In the first condition, only one entire text in the path. The search word is “efficacious” and the search path is "/Users/tianyuyang/Desktop/project2/CSCE6444\_Project2/txt/".

As it is shown as in the figure above, the word “efficacious” is in the Document 1, Line Number is 156 and its line word index is 7. Its word index in its sentence is 25.

In the second condition, there are three divided documents in the path. The search word is “efficacious” and the search path is "/Users/tianyuyang/Desktop/project2/test/".

As it is shown as in the figure above, the word “efficacious” is in the Document 1, Line Number is 156 and its line word index is 7. Its word index in its sentence is 25.

## 3.2 Example for the word “materials”



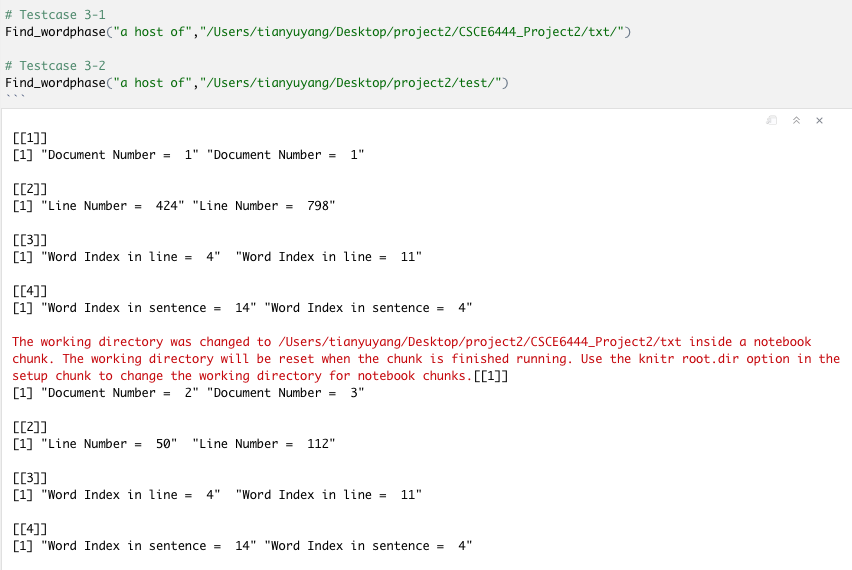
In the first condition, only one entire text in the path. The search word is “materials” and the search path is "/Users/tianyuyang/Desktop/project2/CSCE6444\_Project2/txt/".

As it is shown as in the figure above, there are totally three “materials”. The first word “materials” is in the Document 1, Line Number is 432 and its line word index is 1. Its word index in its sentence is 5. The second word “materials” is in the Document 1, Line Number is 432 and its line word index is 10. Its word index in its sentence is 6. The third word “materials” is in the Document 1, Line Number is 485 and its line word index is 1. Its word index in its sentence is 24.

In the second condition, there are three divided documents in the path. The search word is “materials” and the search path is "/Users/tianyuyang/Desktop/project2/test/".

As it is shown as in the figure above, there are totally three “materials”. The first word “materials” is in the Document 2, Line Number is 58 and its line word index is 1. Its word index in its sentence is 5. The second word “materials” is in the Document 2, Line Number is 58 and its line word index is 10. Its word index in its sentence is 6. The third word “materials” is in the Document 2, Line Number is 111 and its line word index is 1. Its word index in its sentence is 24.

## 3.3 Example for the phrase “a host of”



In the first condition, only one entire text in the path. The search word is “a host of” and the search path is "/Users/tianyuyang/Desktop/project2/CSCE6444\_Project2/txt/".

As it is shown as in the figure above, there are totally two “a host of”. The first phrase “a host of” is in the Document 1, Line Number is 424 and its line word index is 4. Its word index in its sentence is 14. The second phrase “a host of” is in the Document 1, Line Number is 798 and its line word index is 11. Its word index in its sentence is 4.

In the second condition, there are three divided documents in the path. The search word is “a host of” and the search path is "/Users/tianyuyang/Desktop/project2/test/".

As it is shown as in the figure above, there are totally two “a host of”. The first phrase “a host of” is in the Document 2, Line Number is 50 and its line word index is 4. Its word index in its sentence is 14. The second phrase “a host of” is in the Document 3, Line Number is 112 and its line word index is 11. Its word index in its sentence is 4.

# Conclusion

In this project, we have done a lot of work on Natural language processing. We mainly learned and used several packages to process this book. By doing this project, we deepened our understanding of the R and Natural language processing, and we improved our R programming capabilities.

In part a, we tried all functions in lecture 4, and we had a basic understanding of this book.

In part b, we tried to find the 10 longest sentences in this book. First, we got the vectors contains every sentence in the text and then wrote two R functions to realize the function.

In part c. we displayed the dendrogram and the “WordCloud” for each paragraph. “WordCloud” is a very powerful package. We got two kinds of important graphs for text and they are very intuitive and contain a lot of information. From these graphs, we can easily get the frequency of words used in each paragraph. We realize that data visualization is both an art and a science that makes complex data easier to understand and use.

In part d, we found the longest word and longest sentence in each of the paragraphs. We have further enhanced our ability to process texts and have a better understanding of the structure of this article.

In part e, we used a package called "WordNet", it is a large lexical database of English. First, we got the words that contain no less than 5 characters and then we used this package's function to distinguish whether a word is a noun or a verb.

In part f, we used package “zipfR” to analyze word frequency. It has powerful statistical models and utilities for the analysis of word frequency distributions.

In part g, we generated bigrams and trigrams for all words in the first three paragraphs. To do that, we used a function from the package "quanteda". It's a very useful package to manage and analyze textual data.

In part h, we used “corpusTools”, “stringi”, and “quanteda”. We chose three functions from each package and apply them to this book. These packages provide tools that can give us a higher understanding of the text. Also, we tried to understand the theme of the book through the work we did before. This book is mainly about the author's explanations about dreams. It's remarkably efficient to understand what a book is about by some simple applications of NLP.

Further, we have written some functions to search for a specific word or phrase in the documents. This is a very important and common feature that we use every day. Through the implementation of this function in R, our programming ability has been improved.

We believe that Natural language processing is a very important subject. Through it, we can efficiently process some long texts to get a general understanding of its content. Computers are great at working with standardized and structured data like database tables and financial records. But it’s not an easy task teaching machines to understand how we communicate. By using the technology of Natural Language Processing, we can aid computers to understand the human’s natural language.

In the future, we intend to learn more about NLP, especially in areas related to deep learning, because it is very interesting and powerful. Also, we realize that R is easily extensible through functions and extensions, and there are so many useful packages. But it's more important to choose the appropriate package and combine its functionality with your goals.